# Правила и стандарты разработки

# конфигураций и продуктов 1С

# для внешних подрядчиков

**Введение**

**Назначение документа**

* Определение внутреннего корпоративного стандарта разработки конфигураций для платформы «1С: Предприятие 8» в целях улучшения качества производимого программного продукта, оптимизации трудозатрат на поддержку и развитие прикладных решений.
* Текущий документ совместно с методиками и стандартами фирмы 1С является основой для определения нормативных критериев оценки качества кода.

**Подходы к оценке качества кода**

* Оценка качества кода выполняется на основании методик, стандартов фирмы «1С» и правил, изложенных в данном документе.
* Аудит кода проводится как вручную, так и с использованием автоматизированных средств (1C АКП, SonarQube)
* Если программный код не отвечает нормативным требованиям, архитектор Заказчика может запретить поставку в рабочую среду до устранения отклонений.

**Примечания**

* В основу правил положены методические материалы фирмы «1С» - [«Система стандартов и методик разработки конфигураций для платформы «1С: Предприятие 8](https://its.1c.ru/db/v8std)»
* [Общие подходы к рефакторингу](https://refactoring.guru/ru/refactoring)

## Работа с хранилищем (хранилище Заказчика)

* Запрещено захватывать корень более чем на час.
* Любые манипуляции с восстановлением хранилища может делать только администратор.
* Управлять версиями («сливать») также может только администратор.
* В конце каждой недели необходимо проверять захваченные объекты конфигурации.
* При помещении захваченного корня использование рекурсии можно делать только в случае, если захвачены все объекты. В противном случае процедура помещения может значительно увеличить время помещения объектов.
* **При помещении в хранилище необходимо убедиться, что нет регистров без регистраторов, подписок с несуществующими обработчиками, объектов без указания подсистем.**
* **Правила комментирования и помещения изменённых объектов в хранилище описаны в разделе «Правила комментирования».**

### Правила «хорошего тона» или «джентельменские соглашения» при работе с хранилищем:

* перед захватом корня (для добавления новых объектов или снятия с поддержки объекта в конфигурации поставщика) – желательно обновить конфигурацию своей локальной тестовой БД из хранилища, чтобы исключить возникновение ошибок вида «обработчик события не найден» и т.д.;
* корень конфигурации по возможности не держать долго. Требуется действовать по принципу: «Захватили – добавили объект – отпустили». После этого сам новый объект захватывается и с ним ведется дальнейшая работа. Не следует оставлять захваченным корень перед выходными и праздничными днями (только если не планируется работа в эти дни);
* не помещать код и объекты в хранилище до тех пор, пока нет уверенности в том, что вся разрабатываемая функциональность работает и сможет пройти тестирование и проверки (если только правилами конкретного проекта не предусмотрена иная логика помещения и тестирования). Подразумевается, если код помещен в хранилище – значит он готов к внедрению в рабочих базах. Исключение может составить случай, когда кто-то из других разработчиков просит отпустить тот или иной объект. Но даже в этом варианте лучше отпустить объект без помещения его в хранилище (предварительно сохранив свою конфигурацию в отдельный файл). После того, как другой разработчик поработает с объектом и отпустит его – обновить конфигурацию из хранилища, захватить объект и уже в него внести свои изменения через сравнение-объединение с файлом;
* желательно обновлять конфигурацию из хранилища каждый вечер, перед окончанием рабочего дня;
* перед обновлением конфигурации из хранилища также рекомендуется посмотреть историю и проанализировать, не были ли за последнее время изменены или добавлены реквизиты в «большие» таблицы БД (если разрабатываемая БД много «весит»). Если таковые изменения были – следует учесть, что после получения объектов из хранилища начнется реструктуризация информационной базы разработчика и эта операция может продлиться долго и рабочее время будет потеряно впустую. В таком случае лучше запускать обновление из хранилища в конце рабочего дня, чтобы реструктуризация прошла в нерабочие часы;
* если какая-то конкретная доработка приводит к длительной реструктуризации – необходимо «придержать» объект до окончания рабочего дня и оповестить всех коллег-разработчиков о необходимости обновления конфигурации из хранилища по окончании рабочего дня (чтобы не останавливать работу команды);
* при помещении в хранилище регистра, подчиненного регистратору – не забывать помещать также и сам регистратор (частая «ошибка новичка»). Помещаемый регистр должен (по возможности) сразу содержать все нужные измерения и ресурсы. При помещении любого нового объекта не забывать поместить роли для работы с ним. При помещении новой подписки на событие не забывать помещать соответствующий обработчик (можно «пустую» процедуру до окончания разработки);
* при помещении объекта в хранилище необходимо обязательно писать краткий комментарий о том, что было сделано (какая функция/процедура изменена/добавлена и т.д.), см. правила комментирования
* При помещении группы изменений по одной задаче надо выполнять помещение всех изменений одной записью в хранилище, если это возможно. Исключение составляет захват корня хранилища, его желательно помещать как можно скорее.

# Правила комментирования

* Шаблон комментария в модулях
* Правила комментирования в модулях
* Шаблон комментария при помещении в хранилище
* Общие правила комментирования

## Шаблон комментария в модулях

// <АббревиатураКомпании>; <ФамилияИсполнителя> (<?"", ДатаВремя, "ДФ='dd.mm.yyyy'">)); №.<НомерЗадачи> <НазваниеЗадачи> №<НомерЗадачиВнеш>

> НомерЗадачи - Номер задачи из внутреннего трекера.

> НазваниеЗадачи - Название задачи из внутреннего трекера.

> НомерЗадачиВнеш - Номер задачи из трекера заказчика. Указывается в задаче на разработку.

При изменении блоков кода необходимо указывать начало и окончание вносимых изменений. При этом, комментарий блока кода в начале должен начинаться с // +++, а комментарий в конце с // ---.

* // +++НашаФирма; <ФамилияИсполнителя> (<?"", ДатаВремя, "ДФ='dd.mm.yyyy'">)); №.<НомерЗадачи> <НазваниеЗадачи> №<НомерЗадачиВнеш>
* // ---НашаФирма; <ФамилияИсполнителя> (<?"", ДатаВремя, "ДФ='dd.mm.yyyy'">)); №.<НомерЗадачи> <НазваниеЗадачи> №<НомерЗадачиВнеш>

**После символа «//» необходим пробел, иначе автоматизированные системы проверки кода будут указывать на ошибку.**

Пример комментария:

*// +++ НашаФирма; Смирнов; 25.01.2021; №0000176234 №ITMNK-00380*

*// Измененный/Добавленный код*

*// --- НашаФирма; Смирнов; 25.01.2021; №0000176234 №ITMNK-00380*

## Правила комментирования в модулях

* Если выполняется изменение кода типовой конфигурации, то удаляемый код должен быть закомментирован. Если доработки выполняются в нетиповых объектах, неиспользуемый код удаляется.  
  Пример доработки типового модуля:

*//Код до модификации*

*ПеременнаяТиповогоКода = Значение1;*

*ПроцедураТиповогоКода( Значение1, Значение2 );*

*//Код после модификации:*

*ПеременнаяТиповогоКода = Значение1;*

*// +++ НашаФирма; Смирнов; 20.12.17; №0000176234 №ITMNK-00382*

*НоваяПеременная = нфПреобразоватьЗначение(Значение1);*

*//ПроцедураТиповогоКода( Значение1, Значение2 );*

*ПроцедураТиповогоКода( НоваяПеременная, Значение2 );*

*// --- НашаФирма; Смирнов; 20.12.17; №0000176234 №ITMNK-00382*

* Если добавляется новая процедура/функция в существующий модуль, то комментарии к ней необходимо писать в начале процедуры (для каждой процедуры, если их несколько). Ниже строки КонецПроцедуры никаких комментариев не писать! То же самое касается объявления общих переменных в начале модуля. Это необходимо для правильного сравнения-объединения модулей.

*// +++ НашаФирма; Смирнов; 20.12.17; №0000176234 №ITMNK-00382*

*// Заголовок процедуры*

*Процедура НоваяПроцедура()*

*...*

*КонецПроцедуры // ниже этой строки ничего не писать!*

## Шаблон комментария при помещении в хранилище

Измененные объекты помещаются в хранилище в рамках одной задачи. **Не допускается помещать (одной версией) в хранилище изменения, выполненные по разным задачам.**

Шаблон комментария:

НомерЗадачи

НомерЗадачиВнеш

Описание задачи:

НомерЗадачи - Номер задачи из внутреннего трекера.

НомерЗадачиВнеш - Номер задачи из трекера заказчика. Указывается в задаче на разработку.

Описание задачи - краткое описание задачи из трекера.

Пример:

0000176234

ITMNK-00382

Описание задачи: В справочник "ДоговорыКонтрагентов" добавить реквизит "нфСлужебный" Тип....

## Общие правила комментирования

* Простой код необходимо писать и структурировать так, чтобы никаких дополнительных пояснений к нему не требовалось.
* Комментарии, которые не поясняют, а пересказывают код, недопустимы.
* Сложный код необходимо сопровождать сжатыми и четкими пояснениями.
* Длинные линейные фрагменты кода при помощи комментариев необходимо разбивать на фрагменты.
* Если код требует комментария для пояснения работы - в первую очередь необходимо рассмотреть варианты рефакторинга, чтобы код не требовал комментария.
* Длина строки комментария не должна превышать 120 символов.
* Заголовки процедур и функций (за исключением предопределенных) являются строго обязательными.

# Общие требования по доработке типовых конфигураций на платформе 1С Предприятие 8

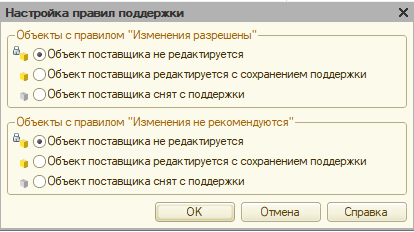
* Модификация типовых конфигураций
  + Подготовка типовой конфигурации перед началом разработки
  + Правила добавления и изменения объектов метаданных
  + Доработка форм
  + Доработка общих модулей, модулей объектов, модулей менеджеров
  + Роли
  + Интерфейс
* Повторное использование кода, стандартные библиотеки.
* Регистрация ошибок, сообщения и предупреждения

## Модификация типовых конфигураций

### Подготовка типовой конфигурации перед началом разработки

После создания хранилища для типовой конфигурации архитектор проекта или ведущий разработчик выполняет следующие действия:

* Включить возможность изменения. В диалоговом окне настройки правил поддержки включить флаги «Объект поставщика не редактируется».

[](https://github.com/kpostolov/dev_rules/blob/master/img/repsettings.png)

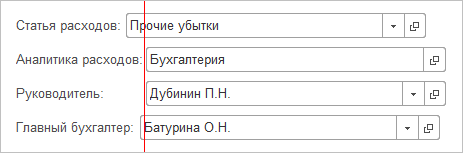
* Изменить режим поддержки для корня конфигурации. Установить «Объект поставщика редактируется с сохранением поддержки».

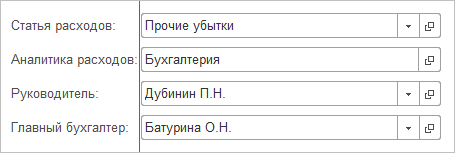
### Правила добавления и изменения объектов метаданных

* Снятие объектов типовой конфигурации с поддержки должно быть согласовано с архитектором
* Все изменения, связанные со схемой данных, выполняются в основной конфигурации. Добавлять реквизиты, объекты метаданных в расширения **запрещено**. Более подробно работа с расширениями описана в разделе **«Применение расширений»**
* Новым объектам, реквизитам, переменным в типовом модуле и т.п. в начале названия добавляем **префикс, идентифицирующий фирму-подрядчика:** **«нф»**, например: **нфСумма**; если добавили табличную часть **нфМастера**, то реквизиты ТЧ пишем без префикса; если добавили функцию **нфФункция**, то локальные переменные пишем без префикса.
* Синоним добавленных объектов не должен содержать префикс «нф». Если синоним совпадает с синонимом типового объекта, то в конец синонима добавлять «(НашаФирма)». Для новых регистров, отчетов, обработок, перечислений, ролей в конце синонима добавлять «(НашаФирма)». Например: «Отчет ДДС (НашаФирма)»
* Все добавленные объекты включаются в подсистему **нфДобавленныеОбъекты**
* Все измененные типовые объекты добавляются в подсистему **нфИзмененныеОбъекты**
* При добавлении отчетов указывать хранилище вариантов, используемое в конфигурации
* При добавлении объектов конфигурации необходимо назначать права на объект. Если в задаче нет дополнительных указаний, права на чтение и запись давать роли **нфБазовыеПрава**.
* При добавлении регламентного задания проверять, что в конфигураторе признак «Использование» = ЛОЖЬ
* У типовых подписок на события не изменять источники данных, вместо этого создавать копию подписки с префиксом «нф» и в неё добавляем свои источники данных.
* Новые справочники, документы и отчеты к типовым механизмам БСП: Свойства, запрет редактирования, версионирование, РЛС, варианты отчетов.
* Объекты метаданных сортируются в дереве конфигурации по имени и по возрастанию.

### Доработка форм

* Основные изменения типовых форм выполняются **программно**. Если есть потребность **значительного** изменения типовой формы, доработка выполняется непосредственно в форме, программно переопределяются события типовых элементов.
* Новым реквизитам и элементам формы добавлять префикс «нф»
* Настройку условного оформления форм и динамических списков необходимо делать в коде формы.
* Настройки условного оформления должны производится при создании формы и потом не должны модифицироваться
* Механизмы управления видимостью и доступностью элементов формы реализуются программно согласно
* Поля реквизитов должны выравниваться по опорной линии.

Неправильно: [](https://github.com/kpostolov/dev_rules/blob/master/img/1a1.png)

Правильно: [](https://github.com/kpostolov/dev_rules/blob/master/img/2a1.png)

### Модули конфигурации

* Необходимо группировать все процедуры/функции в модулях по их назначению с использованием областей. А также с разделением на интерфейс модуля и служебную часть [см. стандарты разработки на ИТС](https://its.1c.ru/db/v8std#content:455:hdoc)
* В модуле менеджера и модуле объекта должна быть прописана директива препроцессора: «#Если Сервер Или ТолстыйКлиентОбычноеПриложение Или ВнешнееСоединение Тогда».
* Добавление общих модулей согласуется с архитектором
* Добавлять суффикс в имени общего модуля в зависимости от свойства модуля Клиент, ВызовСервера, ПовтИсп. Создание новый общих модулей выполнять в соответствии с [правилами создания общих модулей](https://its.1c.ru/db/v8std#content:469:hdoc@5e224c3e)
* Новому серверному модулю устанавливать свойство «Внешнее соединение»
* Для модулей со свойством «Сервер» и «Внешнее соединение» не добавлять свойство «Вызов сервера», а при необходимости создавать новый модуль и из него вызывать функции серверного.
* В типовых модулях конфигурации недопустимо создание функций и процедур. Для этого создается общий модуль, например, если вносятся изменения в модуль объекта документа ПоступлениеТоваровУслуг, то создается общий модуль нфПоступлениеТоваровУслугМодульОбъекта, для типового общего модуля нф<Имя модуля>Расширение, и в нем реализуются необходимые процедуры и функции. В дорабатываемом модуле должны быть только вызовы процедур/функций.
* В дорабатываемом/разрабатываемом модуле должно применяться автоформатирование (alt+shift+F). Разработанный код должен быть выровнен: знаки «=» друг под другом, табуляции, отступы после функциональных блоков - код должен быть удобочитаемым
* Все методы добавленных подписок на события должны вызываться из модуля нфПодпискиНаСобытия. Процедуры модуля нфПодпискиНаСобытия не должны содержать реализацию, а только вызов процедур из других модулей.  
  Например, создали подписку **нфДоговорыКонтрагентовПередЗаписью**, назначаем метод **нфПодпискиНаСобытия.ДоговорыКонтрагентовПередЗаписью**.

Реализация метода:

Процедура ДоговорыКонтрагентовПередЗаписью(Источник, Отказ) Экспорт

нфДоговорыКонтрагентов.ДоговорыКонтрагентовПередЗаписью( Источник, Отказ );

КонецПроцедуры

### Роли

* Типовые роли конфигурации не изменяются. Если есть необходимость доработать типовую роль, создать новую с префиксом нф и выполнить модификацию.
* Все добавленные роли должны делиться на следующие категории:
  + Ассоциированные с должностями (списком обязанностей, конкретным функционалом) сотрудников (нфРуководительОтделаПродаж, нфКладовщик);
  + Ассоциированные с определенной функциональностью в конфигурации – дают права строго на определенные объекты конфигурации (или строго определенный набор прав на объекты, например: нфПравоРедактироватьСправочникНаправления, нфПравоФормироватьПрайсЛисты, нфТолькоПросмотр);
  + Ассоциированные с определённым разделом (блоком) учета (нфПродажи, нфМаркетинг, нфМСФО)

### Интерфейс

* Все добавленные объекты должны быть доступны через интерфейс пользователя.

## Повторное использование кода, стандартные библиотеки.

* При разработке придерживаться принципов, принятых в дорабатываемой конфигурации. Например, если в новом документе реализуется обработка проведения, то структура процедуры должна соответствовать структуре, принятой в конфигурации. Если документ делает движения по регистрам, то должна быть возможность в пользовательском режиме просматривать эти движения, аналогично функционалу дорабатываемой конфигурации
* Перед началом разработки нового функционала провести анализ на предмет возможности повторного использования кода. Например, часть необходимого функционала может быть реализована в БСП или других библиотеках.
* Для расширения типовой функциональности используем механизм типовых переопределяемых общих модулей выявить возможность вставки кода в модулях с постфиксом Переопределяемый.

## Регистрация ошибок, сообщения и предупреждения

* Все сообщения (предупреждения, уведомления) должны быть достаточно информативными и содержательными. Имена объектов конфигурации в сообщениях (предупреждениях, уведомлениях) должны даваться так, как они представлены в пользовательском интерфейсе
* Конфигурация должна выдавать предупреждения с подробными пояснениями перед выполнением процедур, занимающих продолжительное время
* При выдаче в окно сообщений информации, связанной с конкретным объектом информационной базы, должно быть явно указано, какой объект информационной базы вызвал появление сообщения
* При выдаче пользователю вопросов с несколькими вариантами выбора ответа, по умолчанию должен предлагаться ответ, выбор которого вызывает действия, либо наиболее безопасные для информационной базы, либо предусматривающие контроль пользователя за выполнением действий.
* Модальные диалоги, вопросы, предупреждения не должны вызываться внутри транзакций записи и проведения.
* Для вывода в окно сообщений требуется использовать метод ОбщегоНазначения.СообщитьПользователю
* В тексте сообщений об ошибках или информационных объекты ИБ должны быть выделены символами «<>». Пример текста ошибки: «По договору <№РП122345 от 21.10.2018> задолженность <25'4567.00 руб.> превышает допустимую <1’000.00 руб.>»
* Если используется конструкция **«Попытка Исключение»**, при возникновении ошибки, делать запись в журнал регистрации, с указанием текста ошибки.
* При регистрации ошибки в журнале регистраций не следует использовать функцию «ОписаниеОшибки», т.к. она неинформативна для разработчика, потому что не возвращает стек в тексте ошибки. Правильно записывать в журнал регистрации подробное представление исключения, а краткое представление добавлять в текст сообщения пользователю:

*&НаСервере*

*Процедура ВыполнитьОперацию()*

*Попытка*

*// код, приводящий к вызову исключения*

*....*

*Исключение*

*// Запись события в журнал регистрации для системного администратора.*

*нфОбщегоНазначения.ИмяСобытияЖурналаРегистрацийЗаполнениеТЧТовары(),*

*УровеньЖурналаРегистрации.Ошибка,,,*

*ПодробноеПредставлениеОшибки(ИнформацияОбОшибке()));*

*ВызватьИсключение;*

*КонецПопытки;*

*КонецПроцедуры*

*&НаКлиенте*

*Процедура КомандаВыполнитьОперацию()*

*Попытка*

*ВыполнитьОперацию();*

*Исключение*

*ТекстСообщения = КраткоеПредставлениеОшибки(ИнформацияОбОшибке());*

*ПоказатьПредупреждение(,НСтр("ru = 'Операция не может быть выполнена по причине:'") + Символы.ПС + ТекстСообщения);*

*КонецПопытки;*

*КонецПроцедуры*

# Правила оформления кода

## Оформление модулей

* Длина строки в общем случае не должна превышать 120 символов.
* Для отступов необходимо использовать символы табуляции.
* Одна строка кода - одна управляющая конструкция. Следует избегать однострочных конструкций, содержащих сложную логику

*// Неправильно:*

*Если ЭтоБрак Тогда Продолжить; КонецЕсли;*

*// Правильно:*

*Если ЭтоБрак Тогда*

*Продолжить;*

*КонецЕсли;*

*// Неправильно:*

*Возврат Запрос.Выполнить().Выгрузить().Свернуть("Ссылка").ВыгрузитьКолонку("Ссылка")*

*//Правильно*

*РезультатЗапроса = Запрос.Выполнить();*

*ДанныеЗапроса = РезультатЗапроса.Выгрузить();*

*Возврат ДанныеЗапроса.Свернуть("Ссылка").ВыгрузитьКолонку("Ссылка")*

* Программный код любого метода не должен превышать 200 строк. Для этого следует определить задачи и подзадачи, которые он выполняет, и попробовать вынести подзадачи в отдельный метод Исключения: методы, которые возвращают тексты запросов или какие-либо другие объемные строковые фрагменты
* Использование областей внутри процедуры или функции недопустимо
* Текст запроса должен быть оформлен отдельной функцией (при необходимости переиспользования)
* Следует отделять друг от друга пробелами ключевые слова, вызовы процедур и функций, параметры процедур и функций внутри скобок, операторы.

*// Неправильно:*

*Сообщить("Сумма: "+Сумма);*

*// Правильно:*

*Сообщить("Сумма: " + Сумма);*

* Для разделения на логические части внутри модуля следует использовать пустые строки
* При вызове функции с несколькими параметрами при переносе строк необходимо выравнивать параметры по первому

*// Начальное состояние (строка слишком длинная):*

*НалоговыйУчет.ОстаткиВременныхРазниц(СтрокаВидАктиваОбязательства, СписокОрганизаций, Реквизиты.НачалоГода, Реквизиты.КонДата);*

*// Неправильно:*

*НалоговыйУчет.ОстаткиВременныхРазниц(*

*СтрокаВидАктиваОбязательства, СписокОрганизаций, Реквизиты.НачалоГода, Реквизиты.КонДата);*

*// Лучше:*

*НалоговыйУчет.ОстаткиВременныхРазниц(СтрокаВидАктиваОбязательства,*

*СписокОрганизаций,*

*Реквизиты.НачалоГода,*

*Реквизиты.КонДата*

*);*

*// Правильно:*

*НалоговыйУчет.ОстаткиВременныхРазниц(*

*СтрокаВидАктиваОбязательства,*

*СписокОрганизаций,*

*Реквизиты.НачалоГода,*

*Реквизиты.КонДата*

*);*

* Выравнивание однотипных операторов. При следовании друг за другом нескольких однотипных операторов допускается их выравнивание. Выравнивание следует выполнять с помощью пробелов

*// Правильно:*

*НоваяСтрока = ВидыОпераций.Добавить();*

*НоваяСтрока.ВидОперации = ВидОперации;*

*НоваяСтрока.НомерГруппы = ГруппаПоВидуОперации(ВидОперации);*

*НоваяСтрока.ПоОрганизацииВЦелом = ГруппаПоОрганизации(НоваяСтрок);*

* Управляющие конструкции не должны быть вложены слишком глубок (цикломатическая сложность). Уровень вложенности должен быть менее 5.

Вложенные операторы «Если», «Для», «Для Каждого», «Пока» и «Попытка» являются ключевыми ингредиентами для создания так называемого «спагетти-кода».

Неправильно:

*Если Чтото Тогда // Допустимо - уровень = 1*

*/\* ... \*/*

*Если ЧтоТоЕще Тогда // Допустимо - уровень = 2*

*/\* ... \*/*

*Для Ном = 0 По 10 Цикл // Допустимо - уровень = 3*

*/\* ... \*/*

*Если ОпятьУсловие Тогда // Допустимо - уровень = 4, лимит достигнут, но не превышен*

*Если ЕщеЧтото Тогда // Уровень = 5, Превышен лимит*

*/\* ... \*/*

*КонецЕсли;*

*Возврат;*

*КонецЕсли;*

*КонецЦикла;*

*КонецЕсли;*

*КонецЕсли;*

* Когнитивная сложность метода должна быть менее 15

Когнитивная сложность показывает на сколько сложно воспринимать написанный код. Высокая когнитивная сложность явно указывает на необходимость проведения рефакторинга кода для облегчения его будущей поддержки. Наиболее эффективным способом снижения когнитивной сложности является декомпозиция кода, дробление методов на более простые, а также оптимизация логических выражений.

Ниже приведены правила анализа когда, условия повышения когнитивной сложности:

*// Цикл `Для каждого`*

*Для каждого Элемент Из Коллекция Цикл // +1*

*КонецЦикла;*

*// Цикл `Для`*

*Для Ит = Начало По Конец Цикл // +1*

*КонецЦикла;*

*// Цикл `Пока`*

*Пока Условие Цикл // +1*

*КонецЦикла;*

*// Условие*

*Если Условие Тогда // +1*

*// Альтернативная ветвь условия*

*ИначеЕсли Условие2 Тогда // +1*

*// Ветвь по-умолчанию*

*Иначе*

*КонецЕсли;*

*// Тернарный оператор*

*Значение = ?(Условие, ЗначениеИстина, ЗначениеЛожь); // +1*

*Попытка*

*// Обработка исключения*

*Исключение // +1*

*КонецПопытки;*

*// Переход на метку*

*Перейти ~Метка; // +1*

*// Бинарные логические операции*

*Пока Условие ИЛИ Условие2 Цикл // +2*

*КонецЦикла;*

*Если Условие И Условие2 Тогда // +2*

*ИначеЕсли Условие2 // +1*

*ИЛИ Условие3 И Условие4 Тогда // +2*

*КонецЕсли;*

*Значение = ?(Условие ИЛИ Условие2 ИЛИ НЕ Условие3, // +3*

*ЗначениеИстина, ЗначениеЛожь);*

*Значение = Одно ИЛИ Второе; // +1*

*Значение = А <> B; // +1*

Ниже на примерах кода произведен рассчет когнитивной сложности методов.

*Функция Пример1(ТипКласса)*

*Если ТипКласса.Неизвестен() Тогда // +1, условие, //вложенности нет*

*Возврат Символы.НеизвестныйСимвол;*

*КонецЕсли;*

*НеизвестностьНайдена = Ложь;*

*СписокСимволов = ТипКласса.ПолучитьСимвол().Потомки.Поиск("имя");*

*Для Каждого Символ Из СписокСимволов Цикл // +1, цикл, вложенности нет*

*Если Символ.ИмеетТип(Символы.Странное) // +2, условие вложенное в цикл, вложенность 1*

*И НЕ Символы.Экспортный() Тогда // +1, логическая перация, вложенность не учитывается*

*Если МожноПереопределить(Символ) Тогда // +3, вложенное условие, вложенность 2*

*Переопредялемость = ПроверитьПереопределяемость(Символ, ТипКласса);*

*Если Переопределяемость = Неопределено Тогда // +4, вложенное условие, вложенность 3*

*Если НЕ НеизвестностьНайдена Тогда // +5, вложенное условие, вложенность 4*

*НеизвестностьНайдена = Истина;*

*КонецЕсли;*

*ИначеЕсли Переопределяемость Тогда // +1, альтернативная ветвь условия, вложенность не учитывается*

*Возврат Символ;*

*КонецЕсли;*

*Иначе // +1, ветвь по-умолчанию, вложенность не учитывается*

*Продолжить;*

*КонецЕсли;*

*КонецЕсли;*

*КонецЦикла;*

*Если НеизвестностьНайдена Тогда // +1, вложенности нет*

*Возврат Символы.НеизвестныйСимвол;*

*КонецЕсли;*

*Возврат Неопределено;*

*КонецФункции*

*Функция Пример2(Документ)*

*НачатьТранзакцию();*

*НадоПровести = ?(Документ.Проведен, ЛОЖЬ, // +1, тернарный оператор*

*?(Документ.ПометкаУдаления, ЛОЖЬ, ИСТИНА)); // +2, вложенный тернарный оператор, вложенность 1*

*Попытка // +0, попытка, повышает уровень вложенности*

*ДокументОбъект = Документ.ПолучитьОбъект();*

*Если ДокументОбъект.Проведен Тогда // +2, вложенное условие, вложенность 1*

*Для Каждого СтрокаТабличнойЧасти Из ДокументОбъект.ТабличнаяЧасть Цикл // +3, вложенный цикл, вложенность 2*

*Если СтрокаТабличнойЧасти.Колонка1 = 7 // +4, вложенное условие, вложенность 3*

*ИЛИ СтрокаТабличнойЧасти.Колонка2 = 7 Тогда // +1, логическая операция, вложенность не учитывается*

*Продолжить;*

*КонецЕсли;*

*Если СтрокаТабличнойЧасти.Колонка4 > 1 Тогда // +4, вложенное условие, вложенность 3*

*Прервать;*

*Иначе // +1, ветвь по-умолчанию, вложенность не учитывается*

*Если СтрокаТабличнойЧасти.Колонка1 + СтрокаТабличнойЧасти.Колонка2 = 2 Тогда // +5, вложенное условие, вложенность 4*

*СтрокаТабличнойЧасти.Колонка10 = СтрокаТабличнойЧасти.Колонка1 \* 2;*

*КонецЕсли;*

*КонецЕсли;*

*КонецЦикла;*

*Иначе // +1, ветвь по-умолчанию, вложенность не учитывается*

*НадоПровести = ДокументОбъект.Дата > ТекущаяДата(); // +1, логическая операция, вложенность не учитывается*

*Перейти ~Метка; // +1, переход на метку, вложенность не учитывается*

*КонецЕсли;*

*Если НадоПровести Тогда // +2, вложенное условие, вложенность 1*

*ДокументОбъект.Записать(РежимЗаписиДокумента.Проведение);*

*ИначеЕсли НЕ НадоПровести Тогда // +1, альтернативная ветвь, вложенность не учитывается*

*ДокументОбъект.Записать(РежимЗаписиДокумента.Запись);*

*Иначе // +1, ветвь по-умолчанию, вложенность не учитывается*

*ВызватьИсключение "Как так-то?";*

*КонецЕсли;*

*Исключение // +1, обработка исключения*

*ПовторнаяЗапись = ЛОЖЬ;*

*Попытка // +0, попытка, повышает уровень вложенности*

*Если ДокументОбъект.Проведен Тогда // +3, вложенное условие, вложенность 2*

*ДокументОбъект.Записать(РежимЗаписиДокумента.Запись);*

*КонецЕсли;*

*Исключение // +2, обработка исключения, вложенность 1*

*ПовторнаяЗапись = ИСТИНА;*

*КонецПопытки;*

*Если Не ПовторнаяЗапись Тогда // +2, вложенное условие, вложенность 1*

*Пока ТранзакцияАктина() Цикл // +3, вложенный цикл, вложенность 2*

*ОтменитьТранзакцию();*

*КонецЦикла;*

*КонецЕсли;*

*ВызватьИсключение "Ошибка"*

*КонецПопытки;*

*~Метка:*

*Возврат Неопределено;*

*КонецФункции*

### Методы

* Процедура или функция должна содержать текстовую аннотацию, раскрывающую назначение и особенности эксплуатации метода. Также в аннотации должны быть перечислены и описаны все параметры метода и все варианты возвращаемого значения, если метод технически является функцией.

Комментарий размещается перед объявлением процедуры (функции) и имеет следующий вид:

Секция «Описание» содержит описание назначения процедуры (функции), достаточное для понимания сценариев ее использования без просмотра ее исходного кода. Также может содержать краткое описание принципов работы и перекрестные ссылки на связанные процедуры и функции.

Может быть единственной секцией для процедур без параметров. Описание не должно совпадать с именем процедуры (функции). Для процедур и функций секция должна начинаться с глагола. Для функций это, как правило: «Возвращает…». В тех случаях, когда возвращаемый результат является не основным в работе функции, – то с основного действия, например: «Проверяет…», «Сравнивает…», «Вычисляет…» и т.п. Не рекомендуется начинать описание с избыточных слов «Процедура...», «Функция...», а также с имени самой процедуры (функции), от удаления которых смысл не меняется.

Неправильно:

*// Конструктор объекта WSПрокси.*

*// ...*

*Функция WSПрокси(ПараметрыПрокси) Экспорт*

*// Функция СтрокаТаблицыЗначенийВСтруктуру создает структуру со свойствами, соответствующими...*

*Функция СтрокаТаблицыЗначенийВСтруктуру(СтрокаТаблицыЗначений) Экспорт*

Правильно:

*// Создает прокси на основе определения веб-сервиса и связывает*

*// его с точкой подключения веб-сервиса.*

*// В дополнении к платформенному конструктору Новый WSПрокси:*

*// - включает в себя вызов конструктора WSОпределения;*

*// - на время сеанса кэширует файл WSDL для оптимизации частых обращений к веб-сервису;*

*// - не требует явного указания ИнтернетПрокси (он подставляется автоматически, если настроен);*

*// - выполняет быструю проверку доступности веб-сервиса с помощью операции Ping.*

*// ...*

*Функция WSПрокси(ПараметрыПрокси) Экспорт*

*// Создает структуру со свойствами, соответствующими...*

*Функция СтрокаТаблицыЗначенийВСтруктуру(СтрокаТаблицыЗначений) Экспорт*

Секция «Параметры» описывает параметры процедуры (функции). Если их нет, секция пропускается. Предваряется строкой «Параметры:», затем с новой строки размещаются описания всех параметров.

Имя параметра необходимо стремиться выбирать таким образом, чтобы его назначение было понятно в контексте функции без дополнительных пояснений

Описание типа является обязательным. Тип может быть описан явно, при этом может быть указан или один тип или список типов. Под «списком типов» подразумеваются имена типов, разделенные запятыми. Имя типа может быть простым (в одно слово) или составным - в два слова, разделенных точкой. Например: Строка, Структура, Произвольный, СправочникСсылка.Сотрудники.

В качестве типов значений следует использовать только существующие в платформе типы, а также специальные типы: ОпределяемыйТип.<Имя>, СправочникСсылка, ОбъектМетаданныхОтчет, РасширениеДекорацииФормыДляНадписи и т.п.

Неправильно:

*// КоллекцияСтрок - КоллекцияЗначений – коллекция для сравнения;*

*// ФормируемыйОтчет - ОбъектМетаданных: Отчет*

*// ПрисоединенныйФайлОбъект - элемент справочника файлов.*

Правильно:

*// КоллекцияСтрок – ТаблицаЗначений, Массив, СписокЗначений – коллекция для сравнения.*

*// ФормируемыйОтчет – ОбъектМетаданныхОтчет*

*// ПрисоединенныйФайлОбъект - ОпределяемыйТип.ПрисоединенныйФайлОбъект - элемент справочника файлов.*

Текстовое описание параметра рекомендуется заполнять в том случае, когда только имени параметра в контексте функции не достаточно для понимания его назначения, либо требуется дать дополнительную информацию о типе, поясняющие назначение параметра, а также может приводиться наглядный пример с ожидаемым значением параметра.

Для параметров типа Структура и ТаблицаЗначений также задается описание их свойств и колонок, которые начинаются с новой строки и предваряются символом \*.

Например:

*// Параметры:*

*// СтатусыСерий - ТаблицаЗначений:*

*// \* Серия - СправочникСсылка.СерииНоменклатуры - если серия указана и она может*

*// использоваться с новым значением номенклатуры на указанном складе,*

*// то возвращается переданное значение; если нет - пустая ссылка*

*// \* СтатусУказанияСерий - Число - если серии указываются в ТЧ "Товары", то*

*// возвращается рассчитанный статус, если для переданной*

*// номенклатуры/склада серии не используется - возвращается 0*

*// иначе возвращается переданный статус.*

Для параметров типа Массив следует указывать тип элементов с помощью ключевого слова «из»

В описании массивов, структур и таблиц значений могут быть вложенные описания, при этом перед именами вложенных свойств число звездочек увеличивается: для первого уровня вложенности 2 звездочки, для второго 3 и т.д.

// Параметры:

// СведенияОбОбновлении - Массив из Структура:

// \* КодАдресногоОбъекта - Структура:

// \*\* КодРегиона - Число - код региона (длина - 2).

// \*\* КодНаселенногоПункта - Число - код населенного пункта (длина - 3).

// \*\* КодУлицы - Число - код улицы (длина - 4).

// \* Наименование - Строка

// \* ОбновлениеДоступно - Булево

//

Секция «Возвращаемое значение» описывает тип и содержание возвращаемого значения функции. Для процедур эта секция отсутствует. Предваряется строкой «Возвращаемое значение:». Затем с новой строки тип возвращаемого значения, дефис и текст описания. При использовании возвращаемого значения составного типа следует каждый тип писать с новой строки и с дефиса.

*// Возвращаемое значение:*

*// Строка*

*// Возвращаемое значение:*

*// Булево - Истина, если хотя бы одна из переданных ролей доступна текущему пользователю, либо у него есть административные права.*

*// Возвращаемое значение:*

*// - ЛюбаяСсылка - ссылка на предопределенный элемент.*

*// - Неопределено - если предопределенный элемент есть в метаданных, но не создан в ИБ.*

*// Возвращаемое значение:*

*// - СправочникСсылка.Пользователи*

*// - СправочникСсылка.ВнешниеПользователи*

* Параметры любой процедуры или функции подразделяются на основные и дополнительные. Основные параметры передаются классическим образом, как параметры, дополнительные параметры передаются в виде структуры. Количество основных параметров любого метода не должно превышать трех. Структура дополнительных параметров таким образом является четвертым параметром. Для первоначальной инициализации структуры дополнительных параметров необходимо создать функцию конструктор. Описание ключей структуры должно быть в заголовке процедуры или функции.

Пример функции-конструктора параметров в модуле ЦенообразованиеКлиентСервер:

*Функция ПараметрыЗаполненияЦеныВСтрокеТЧ() Экспорт*

*ПараметрыЗаполненияЦен = Новый Структура;*

*ПараметрыЗаполненияЦен.Вставить("Дата");*

*ПараметрыЗаполненияЦен.Вставить("Валюта");*

*ПараметрыЗаполненияЦен.Вставить("ПересчитыватьСумму", Истина);*

*ПараметрыЗаполненияЦен.Вставить("ОбязательныеПараметры","Дата,Валюта"); // обязательные параметры, которые нужно заполнять*

*Возврат ПараметрыЗаполненияЦен;*

*КонецФункции*

Имена свойств структуры соответствуют параметрам вызываемой функции. При этом параметры со значениями по умолчанию должны быть явно проинициализированы в этой структуре.

В вызывающем коде не следует инициализировать структуру параметров или добавлять в нее какие-либо другие свойства. Во избежание неоднозначности и скрытых ошибок все допустимые параметры вызываемой функции должны быть определены явно в функции-конструкторе параметров.

Параметр функции не должен возвращать значение. Иными словами не используйте входные параметры функций как дополнительный вывод. Весь вывод должен быть в возвращаемом значении. Если нужно возвращать несколько значений следует использовать такие типы как Структура, Массив и т.д.

*// Плохо:*

*URLСервиса = "";*

*ИмяПользователя = "";*

*ПарольПользователя = "";*

*ЗаполнитьПараметрыПодключения(URLСервиса, ИмяПользователя, Пароль);*

*// Хорошо:*

*ПараметрыПодключения = ПолучитьПараметрыПодключения();*

*// Возвращаемое значение - структура:*

*// URLСервиса - Строка*

*// ИмяПользователя - Строка*

*// ПарольПользователя - Строка*

### Условия

* Предпочтительней использовать тернарный оператор для простых конструкций.

*// Неправильно:*

*Если НДС0 Тогда*

*Возврат 0;*

*Иначе*

*Возврат 18;*

*КонецЕсли;*

*// Правильно:*

*Возврат ?(НДС0, 0, 18);*

* Не допускайте использования вложенных тернарных операторов.
* Ключевое слово Тогда пишется на той же строке, что и последнее условие.
* Сложные условия (содержащие 3 конструкции и более) необходимо выносить в отдельные методы.

*// Неправильно:*

*Если ИдентификаторОбъекта = "АнализСубконто"*

*ИЛИ ИдентификаторОбъекта = "АнализСчета"*

*ИЛИ ИдентификаторОбъекта = "ОборотноСальдоваяВедомость"*

*ИЛИ ИдентификаторОбъекта = "ОборотноСальдоваяВедомостьПоСчету"*

*ИЛИ ИдентификаторОбъекта = "ОборотыМеждуСубконто"*

*ИЛИ ИдентификаторОбъекта = "ОборотыСчета"*

*ИЛИ ИдентификаторОбъекта = "СводныеПроводки"*

*ИЛИ ИдентификаторОбъекта = "ГлавнаяКнига"*

*ИЛИ ИдентификаторОбъекта = "ШахматнаяВедомость" Тогда*

*ПараметрыРасшифровки.Вставить("ОткрытьОбъект", Ложь);*

*ЕстьПоказатель = Ложь;*

*ЕстьКорЗначение = Ложь;*

*ЕстьСчет = Истина;*

*Счет = Неопределено;*

*ПервыйЭлемент = Неопределено;*

*КонецЕсли;*

*// Правильно:*

*Если ОткрыватьОбъектПриИдентификаторе(ИдентификаторОбъекта) Тогда*

*ПараметрыРасшифровки.Вставить("ОткрытьОбъект", Ложь);*

*ЕстьПоказатель = Ложь;*

*ЕстьКорЗначение = Ложь;*

*ЕстьСчет = Истина;*

*Счет = Неопределено;*

*ПервыйЭлемент = Неопределено;*

*КонецЕсли;*

*//Вариант - 1*

*Функция ОткрыватьОбъектПриИдентификаторе(ИдентификаторОбъекта)*

*Возврат ИдентификаторОбъекта = "АнализСубконто"*

*ИЛИ ИдентификаторОбъекта = "АнализСчета"*

*ИЛИ ИдентификаторОбъекта = "ОборотноСальдоваяВедомость"*

*ИЛИ ИдентификаторОбъекта = "ОборотноСальдоваяВедомостьПоСчету"*

*ИЛИ ИдентификаторОбъекта = "ОборотыМеждуСубконто"*

*ИЛИ ИдентификаторОбъекта = "ОборотыСчета"*

*ИЛИ ИдентификаторОбъекта = "СводныеПроводки"*

*ИЛИ ИдентификаторОбъекта = "ГлавнаяКнига"*

*ИЛИ ИдентификаторОбъекта = "ШахматнаяВедомость";*

*КонецФункции*

*//Вариант - 2*

*Функция ОткрыватьОбъектПриИдентификаторе(ИдентификаторОбъекта)*

*ИспользуемыеИдентификаторы = Новый Массив;*

*ИспользуемыеИдентификаторы.Добавить("АнализСубконто");*

*ИспользуемыеИдентификаторы.Добавить("АнализСчета");*

*ИспользуемыеИдентификаторы.Добавить("ГлавнаяКнига");*

*ИспользуемыеИдентификаторы.Добавить("ОборотноСальдоваяВедомость");*

*ИспользуемыеИдентификаторы.Добавить("ОборотноСальдоваяВедомостьПоСчету");*

*ИспользуемыеИдентификаторы.Добавить("ОборотыМеждуСубконто");*

*ИспользуемыеИдентификаторы.Добавить("ОборотыСчета");*

*ИспользуемыеИдентификаторы.Добавить("СводныеПроводки");*

*ИспользуемыеИдентификаторы.Добавить("ШахматнаяВедомость");*

*Возврат ИспользуемыеИдентификаторы.Найти(ИдентификаторОбъекта) <> Неопределено;*

*КонецФункции*

* Избегайте использования больших блоков кода внутри операторов Если. Вместо этого выносить код реализации в отдельный метод.

Неправильно:

*Пока ВыборкаЗапроса.Слудующий() Цикл*

*Если ДанныеВыборкиКорректны( ВыборкаЗапроса ) Тогда*

*//... несколько стрниц кода*

*Иначе*

*//... несколько стрниц кода*

*КонецЕсли;*

*КонецЦикла;*

Правильно:

*Пока ВыборкаЗапроса.Слудующий() Цикл*

*Если ДанныеВыборкиКорректны( ВыборкаЗапроса ) Тогда*

*ВывестиДанныеВыборки( ВыборкаЗапроса )*

*Иначе*

*ДобавитьОшибкуВЛог( ВыборкаЗапроса )*

*КонецЕсли;*

*КонецЦикла;*

* Избегайте использование «Йода-синтаксиса».

// Неправильно:

Если 0 = Сумма Тогда

// Правильно:

Если Сумма = 0 Тогда

### Переменные

* Перечень строковых величин располагать в алфавитном порядке

Неправильно:

*ПараметрыОткрытия = Новый Структура( "Ссылка, Комментарий, Наименование, ПредставлениеУсловий*

*|ПометкаУдаления, Пордяок, ГруппаТомов, НавигационнаяСсылка*

*|УникальныйИдентификатор, Ответственный" ):*

*Правильно - вариант 1:*

*ПараметрыОткрытия = Новый Структура( "ГруппаТомов, Комментарий, НавигационнаяСсылка,Наименование, |Ответственный, ПометкаУдаления, Пордяок, ПредставлениеУсловий,*

*|Ссылка, УникальныйИдентификатор" ):*

Правильно - вариант 2 лучше:

*ПараметрыОткрытия = Новый Структура;*

*ПараметрыОткрытия.Вставить("ГруппаТомов");*

*ПараметрыОткрытия.Вставить("Комментарий");*

*ПараметрыОткрытия.Вставить("НавигационнаяСсылка");*

*ПараметрыОткрытия.Вставить("Наименование");*

*ПараметрыОткрытия.Вставить("Ответственный");*

*ПараметрыОткрытия.Вставить("ПометкаУдаления");*

*ПараметрыОткрытия.Вставить("Порядок");*

*ПараметрыОткрытия.Вставить("ПредставлениеУсловий");*

*ПараметрыОткрытия.Вставить("Ссылка");*

*ПараметрыОткрытия.Вставить("УникальныйИдентификатор");*

* Магические числа необходимо заменять на символьные константы

Магические числа — это значения, встречающиеся в коде, но при этом неочевидно, что они означают. Это затрудняет понимание программы и усложняет её рефакторинг.

Неправильно:

*Функция ПодключитьсяКСервису()*

*Возврат МодульПодключенияКСервисуА.Подключиться( 60, Ложь )*

*КонецФункции*

Правильно:

*Функция ПодключитьсяКСервисуА()*

*Таймаут = 60;*

*ВызыватьИсключениеПриОшибке = Ложь;*

*Возврат МодульПодключенияКСервисуА.Подключиться(Таймаут, ВызыватьИсключениеПриОшибке);*

*КонецФункции*

### Имена процедур, функций, переменных

* Следуйте общему подходу именования

// Неправильно:

этобрак, ЭТОБРАК, этоБрак

// Правильно:

ЭтоБрак

* Не используйте отрицание в именах переменных и методов

*// Неправильно:*

*Функция ПроверкаНеПройдена()*

*...*

*Если Не (Условие И Не ПроверкаНеПройдена()) Тогда*

*// Правильно:*

*Функция ПроверкаПройдена()*

*...*

*Если Не Условие И Не ПроверкаПройдена() Тогда*

## Создание, изменение объектов метаданных

* Если составной тип используется многократно, следует использовать объект конфигурации «Определяемый тип». Пример: «Документ резервирования», «Документ партии».

**Применение расширений**

При доработке (адаптации) типовых конфигураций допускается использование расширений СОВМЕСТНО с внесением изменений в конфигурацию. Необходимость использования расширений оговаривается для конкретного проекта. Варианты расширений:

1. Для доработки типовой конфигурации создается **ОДНО** основное расширение на конфигурацию, оно подключается к хранилищу расширений. Далее идет описание правил работы именно с ним;
2. Отдельные расширения для временного решения определенных проблем («патчи», «hotfix»), если недопустимо оперативное изменение основной конфигурации. Они не подключаются к хранилищу;
3. Отдельные расширения для разработки обособленных подсистем.

Основное назначение расширения - упростить последующее обновление доработанной типовой конфигурации до актуальных версий. Если необходимо расширить или заменить типовые события/процедуры, следует руководствоваться следующими приоритетами: А) Реализовать подписку на событие Б) Реализовать в расширении В) Изменить типовой модуль.

Правила и допущения при работе с расширениями при доработке типовой конфигурации описаны ниже.

1. Выполнять доработку типовых форм, модулей, состав подсистем предпочтительно в расширении. При этом в расширение следует выносить только нужные объекты.
2. Добавление новых объектов метаданных (справочники, реквизиты и т.д.) необходимо делать в основной конфигурации! В расширении только изменение типовых модулей, форм, макетов и т.д.

2.1. Новые собственные объекты добавляем в основную конфигурацию и в ней же дорабатываем. Например, новый документ нфПремияКоДнюЭнергетиков и все его реквизиты, формы, макеты и т.д. добавляется в основную конфигурацию.

2.2. Новые реквизиты для типовых объектов необходимо добавлять в основную конфигурацию. Например, если необходимо в типовой документ «Премия» добавить реквизит нфКомментарий, то добавляем реквизит в основную конфигурацию и затем используем его в модулях или при доработке формы в расширении.

2.3. При редактировании в расширении текста запросов с использованием конструктора он «ругается», если всех таблиц запроса нет в расширении. Добавлять объект в расширение только ради этого (чтобы использовать конструктор запросов) **не следует**, особенно, если запрос использует множество объектов. Можно отредактировать текст запроса конструктором в другом модуле (например, во внешней обработке) и потом перенести в расширение. Количество объектов в расширении должно быть минимально необходимым.

1. Приоритетным является использование префиксов &Перед или &После для доработки процедур. При расширении для функций используется единственный префикс &Вместо. Но с его помощью можно реализовать ту же логику работы, как Перед и После с помощью конструкции ПродолжитьВызов() без копирования полного текста процедуры в расширение. Например:

*&Вместо("ОтборСотрудниковДляВыплаты")*

*Функция РасшЗУП\_ОтборСотрудниковДляВыплаты()*

*Параметры = ПродолжитьВызов();*

*// тут реализована логика работы ПОСЛЕ*

*// +++ НашаФирма, Утешев Д.В [11.06.2019] ОПЭ\_232*

*Параметры.Вставить("ВыплатыВПользуФизическихЛиц", Ложь);*

*// +++ НашаФирма, Утешев Д.В [11.06.2019] ОПЭ\_232*

*Возврат Параметры;*

*КонецФункции*

1. Отдельное замечание про доработку в расширении типовых процедур, функций с префиксом «&Вместо» с копированием текста процедуры. Эту возможность следует использовать взвешенно и обоснованно из-за необходимости адаптации этих доработок при обновлении релиза типовой конфигурации. Например, если процедура в расширении значительно переписывается и обновлению уже не подлежит, то можно использовать «&Вместо». В тех случаях, когда требуется внести локальное изменение только в часть процедуры и при этом не меняется основной её алгоритм, то лучше внести изменение в основную конфигурацию. Спорные случаи согласовывать с архитектором. В любом случае все изменения внутри типовых процедур (не важно в расширении или нет) нужно комментировать по общим правилам!
2. Добавление новых ролей необходимо выполнять в основной конфигурации. Типовые Роли в расширении по возможности не дорабатывать (т.к. эти изменения потом сложно анализировать и адаптировать при обновлении конфигурации). Решать задачи добавлением новых ролей основной конфигурации там, где это возможно. Необходимость доработки типовых ролей (не важно в расширении или нет), необходимо согласовывать с архитектором.
3. Доработки расширения разработчиками ведутся в хранилище расширений по правилам, описанным в «Правила работы с хранилищем».
4. Если какая-либо из исходных типовых форм расширена - не следует в расширении злоупотреблять "визуальным конфигурированием" формы (перемещать группы и кнопки, добавлять или удалять элементы и команды). Все изменения расширенной формы следует делать программно (см. раздел «Доработка форм» в «Общие требования по доработке типовых конфигураций на платформе 1С Предприятие 8»).